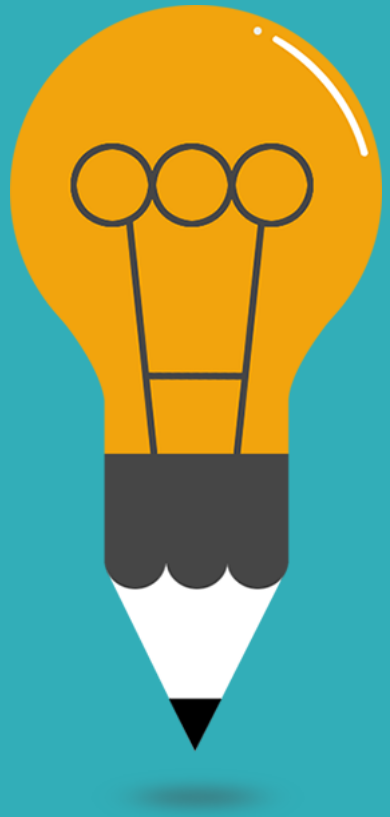
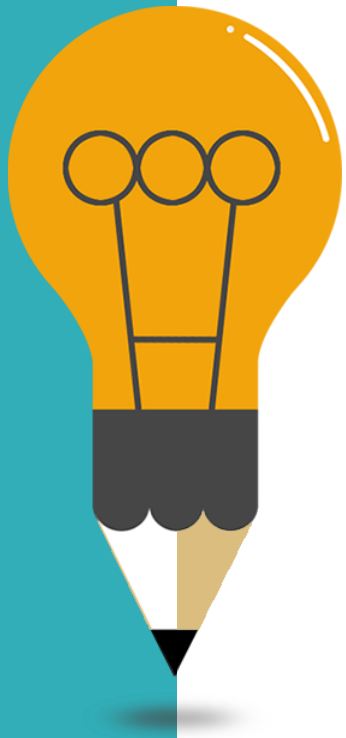


IMDb Score Prediction

2018 Spring ESC Final Project Presentation

김현지, 박광문, 신경준, 유영진 정서형

What's Coming?



01 EDA 요약 및 추가 EDA

02 Model Selection: 회귀분석, 랜덤 포레스트, 부스팅

03 최종 모델 선택 및 결과

04 Further Comments: 모델의 한계, 데이터의 한계

01. EDA 요약 및 추가 EDA

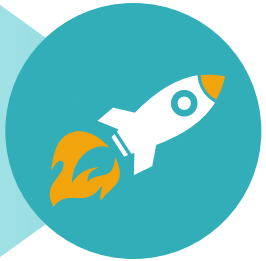
01 삭제한 변수

02 카테고리 변수와 NA처리

03 숫자형 변수와 NA처리

04 파생변수와 NA처리

05 변수 TRANSFORMATION



EDA 작업

Week 1 EDA 정리

데이터 셋: 4943 obs. of 28 variables

```
> names(train_original)
 [1] "color"                     "director_name"
 [3] "num_critic_for_reviews"    "duration"
 [5] "director_facebook_likes"   "actor_3_facebook_likes"
 [7] "actor_2_name"              "actor_1_facebook_likes"
 [9] "gross"                     "genres"
[11] "actor_1_name"              "movie_title"
[13] "num_voted_users"           "cast_total_facebook_likes"
[15] "actor_3_name"              "facenumber_in_poster"
[17] "plot_keywords"             "movie_imdb_link"
[19] "num_user_for_reviews"      "language"
[21] "country"                   "content_rating"
[23] "budget"                    "title_year"
[25] "actor_2_facebook_likes"    "imdb_score"
[27] "aspect_ratio"              "movie_facebook_likes"
```

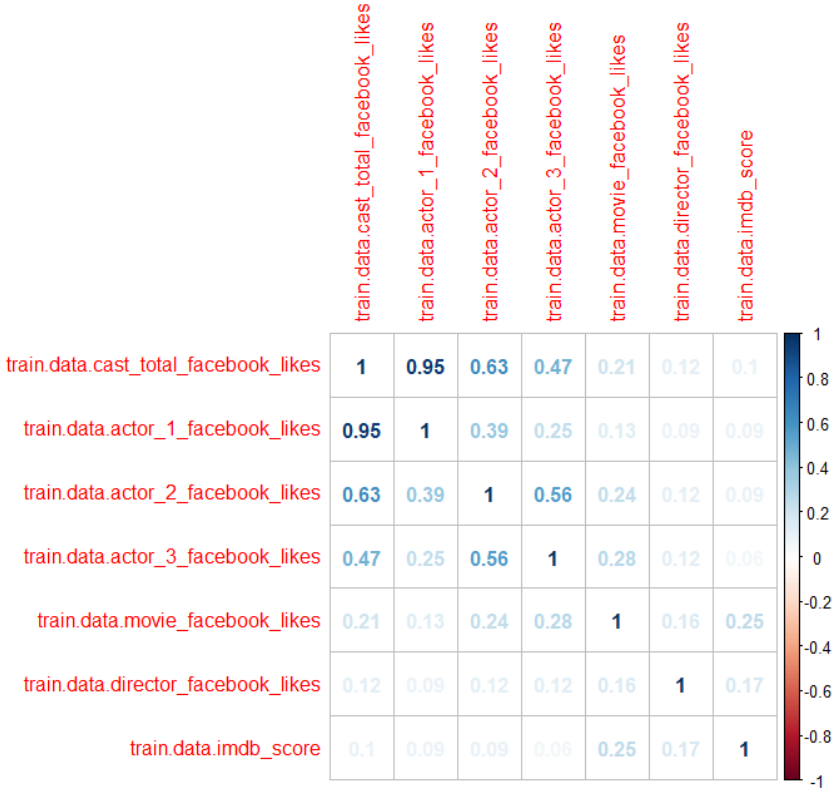
삭제한 변수

파생변수로 활용한 변수

종속변수

01. 삭제한 변수

Facebook like 수와 관련된 변수: actor 1, actor 2, actor 3, director, movie



- actor 페이스북 라이크 수는 imdb score와 상관관계가 상당히 낮은 것으로 보여 삭제
- director 페이스북 라이크 수는 유명 감독일 수록 페이스북 라이크 수가 높은 것으로 나타남
  추후 유명 감독을 나타내는 파생변수 생성으로 정보가 겹칠 것으로 판단하여 삭제
- movie 페이스북 라이크 수는 imdb score와 상관관계를 보이지만 0이 30%를 넘어 data의 imbalance를 초래함으로 삭제

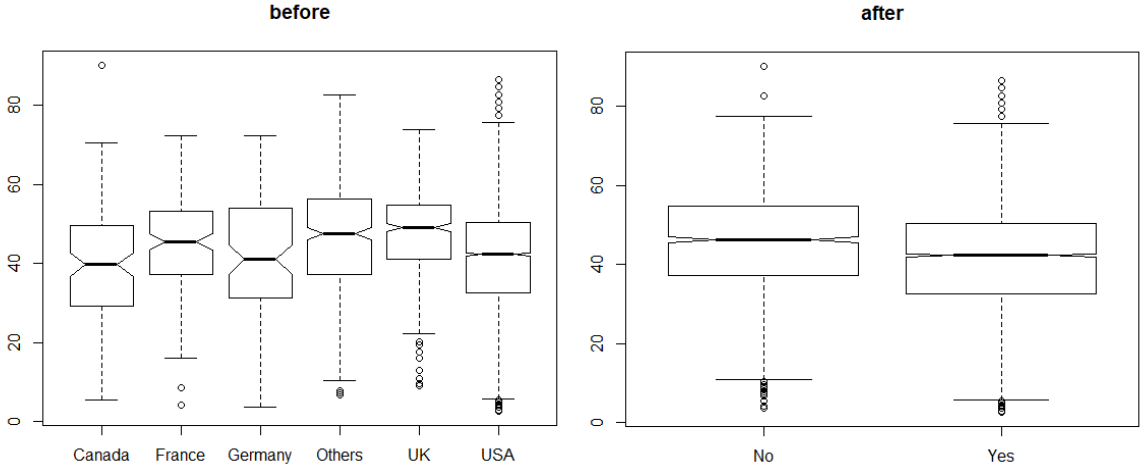
01. 삭제한 변수

- movie title, plot keyword, movie imdb link
  활용이 어렵고 plot keyword의 경우 genre와 겹치는 정보라고 판단하여 삭제

- language
  country 변수와 정보가 겹친다고 판단하여 삭제

02. 카테고리 변수와 NA 처리

country



1. 기존 Country 변수 정리
USA이면 YES, 다른 경우 NO로 처리

2. Main Idea
기존 6개 범주로 나눈 경우, 월등하게 점수가 높은 국가를 찾기 힘들었음
(대다수의 영화가 미국의 것임을 고려해 간략화)

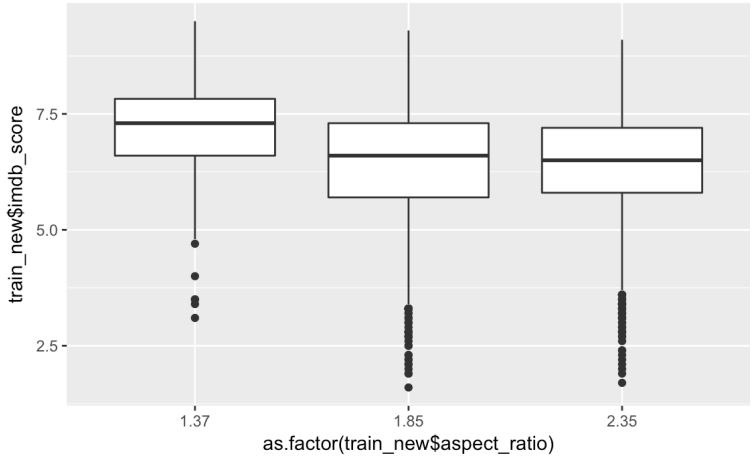
3. Train set에서의 효과
Imdb score를 제곱해 상자그림 대칭화, 후 Notch 비교를 통해 유의미한 차이 확인

02. 카테고리 변수와 NA 처리

aspect_ratio

```r
train_raw$aspect_ratio <- as.factor(train_raw$aspect_ratio)
summary(train_raw$aspect_ratio)
```

```
   1.37                1.85                2.35
1.18  1.2 1.33 1.37 1.44  1.5 1.66 1.75 1.77 1.78 1.85 1.89    2 2.2 2.24
   1    1   67   99    1    2   63    3    1  110 1862    1    5  15    1
2.35 2.39  2.4 2.55 2.76    4   16 NA's
2316   15    3    2    3    5   43  324
```



- 왼쪽과 같이 총 3개의 범주로 카테고리화
  - 4 4:3이 잘못 인코딩 -> 1.37에 포함
  - 16 16:9이 잘못 인코딩 -> 1.85에 포함
  - NA 최빈값인 2.35으로 인코딩

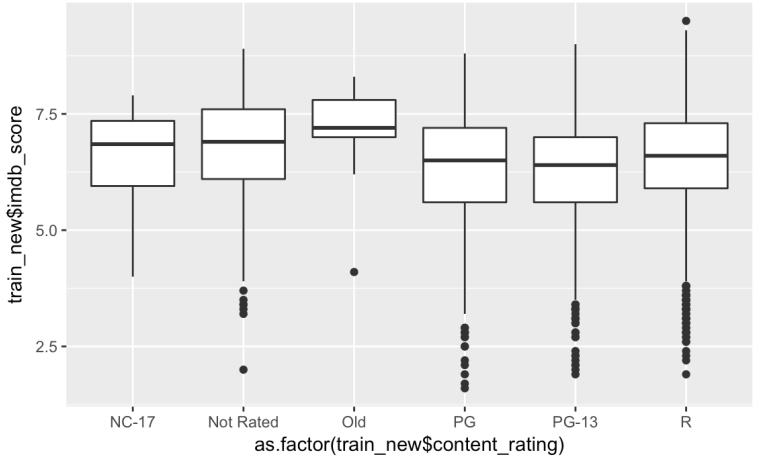
- aspect_ratio가 1.37에 해당하는 경우 rating이 보다 높다. 이는 aspect_ratio가 1.37인 경우 대부분 1940년대 근방 영화들이 많으며, 고전 영화를 선호하는 골수 팬들의 편파적 평점에 의한 것으로 해석됨

02. 카테고리 변수와 NA 처리

content_rating

```r
train_raw$content_rating <- as.factor(train_raw$content_rating)
summary(train_raw$content_rating)
```

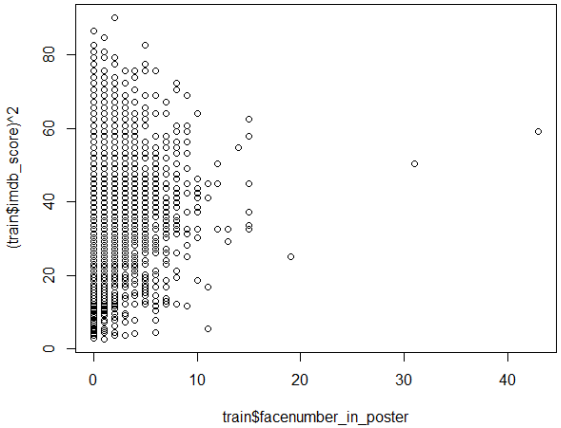
```
 Approved         G        GP         M     NC-17 Not Rated    Passed
       54       110         6         4         7       113         9
       PG     PG-13         R     TV-14      TV-G     TV-MA     TV-PG
      688      1433      2082        29        10        20        11
     TV-Y     TV-Y7  Unrated         X     NA's
        1         1        61        13       291
```

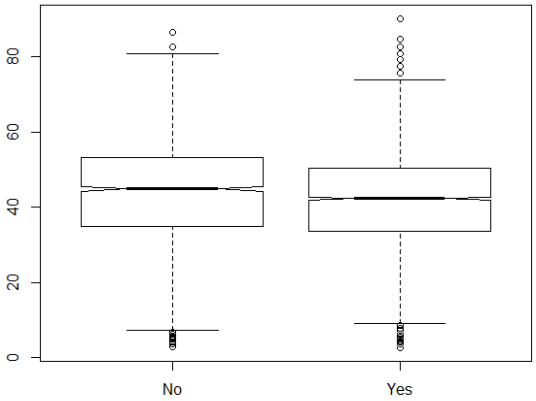


- 미국의 content rating 기준으로 통일
  PG, PG-13, R, NC-17, Not Rated, Old 6개 카테고리로 범주화. Passed, Approved는 1950~60년대 (tv 보급 전) 기준으로 "Old"라는 범주 생성
- NA는 mode인 R로 처리
- 다만 "Passed", "Approved" 였던 영화들이 조금 높은 rating을 받은 것 확인가능: 마찬가지로 클래식 영화 골수 팬들의 영향으로 해석 됨.

02. 카테고리 변수와 NA 처리

facenum_in_poster





1. 기존 facenumber_in_poster 변수 정리
facenumber_in_poster가 0이면 No 부여, 0이 아니면 Yes

2. Main Idea
기존 numeric 변수 facenumber와 평점은 큰 상관이 없음. 따라서 포스터에 얼굴이 존재하냐의 여부로 간략화

3. Train set에서의 효과
Imbd score를 제곱해 상자그림 대칭화 이후 Notch 비교를 통해 유의미한 차이를 확인

02. 카테고리 변수와 NA 처리

genre

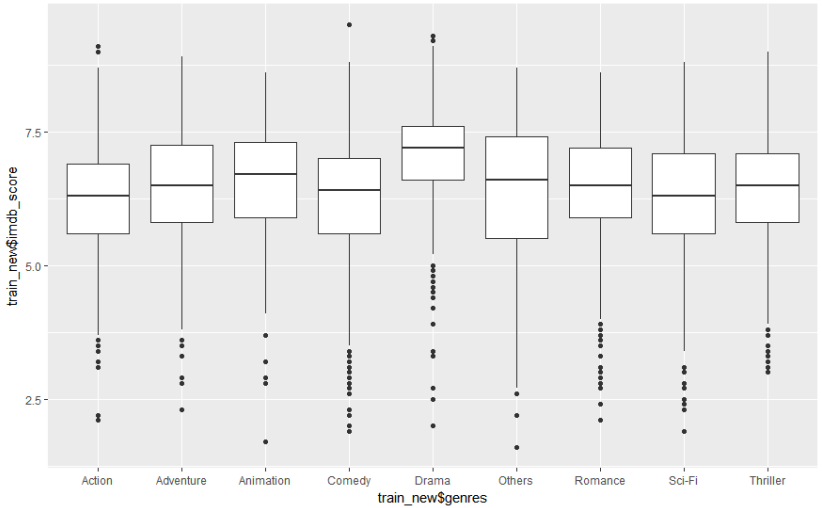
genres
Action|Adventure|Fantasy|Sci-Fi
Action|Adventure|Fantasy
Action|Adventure|Thriller
Action|Thriller
Documentary
Action|Adventure|Sci-Fi
Action|Adventure|Romance
Adventure|Animation|Comedy|Family|Fantasy|Musical|Romance
Action|Adventure|Sci-Fi
Adventure|Family|Fantasy|Mystery

- 총 25 장르, | 로 나뉘어져 있음
- |로 장르를 나누고, 빈도수가 높은 장르를 기준으로 각 obs에 적합한 장르 하나를 부여. 빈도 수가 낮은 장르는 Others로 묶음.dummy variable 처리. 이후 가장 중요하다고 생각되며
  Ex) Drama > Comedy > Thriller > Action > Romance > Adventure > Sci-Fi …
- 각 장르의 빈도 수 계산
- NA 개수: 0

	action	adventure	animation	biography	comedy	crime	document	drama	family	fantasy	history	horror	music	musical	mystery	romance	sci_fi	sport	thriller	war	western	short	reality_tv	film_noir	news
1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
7	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
8	0	1	1	0	1	0	0	0	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
9	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
10	0	1	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
SUM	1134	907	236	288	1830	868	120	2544	533	595	205	557	209	131	486	1081	601	178	1385	210	95	5	2	5	5

02. 카테고리 변수와 NA 처리

genre(continued), color



Genre

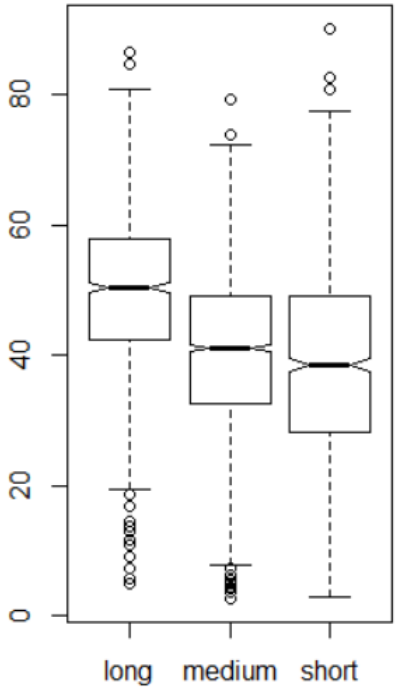
데이터 정리 후 장르별 박스플롯

Color

color 변수는 Color와 Black and White 두 카테고리.
NA에는 color 영화가 비율적으로 훨씬 많은 점을
고려하여 color로 대체

03. 숫자형 변수와 NA 처리

duration



1. 기존 Duration 변수 정리

Five number Summary를 통해 Upper Hinge: 118, Lower Hinge: 93 얻음

if Duration<93 : Short

if 93<Duration<118 : Medium

if 118<Duration : Long

```
> fivenum(train$duration)
[1]   7  93 103 118 511
```

2. Main Idea

Duration이 지나치게 짧거나 긴 영화가 존재

(Series 같은 경우 10편의 총 시간을 합해 놓은 경우가 있음.)

3.Train set에서의 효과

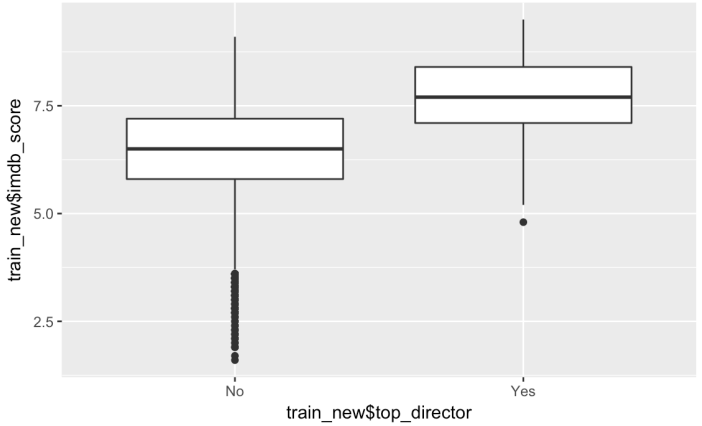
Imbd score를 제곱해 상자그림 대칭화, 후 Notch 비교를 통해 유의미한 차이가 있음을 확인

04. 파생변수와 NA 처리

01/ top_director

top_director에
해당되는 감독 명단

top.directors					
1	Steven Spielberg	18	Sidney Lumet	35	Sergio Leone
2	Martin Scorsese	19	Darren Aronofsky	36	Tony Kaye
3	Ridley Scott	20	Frank Capra	37	Akira Kurosawa
4	Robert Zemeckis	21	George Lucas	38	Irvin Kershner
5	Peter Jackson	22	Jean-Pierre Jeunet	39	Roger Allers
6	David Fincher	23	Jonathan Demme	40	Asghar Farhadi
7	Francis Ford Coppol	24	David Lean	41	Bill Melendez
8	Bryan Singer	25	Frank Darabont	42	Cary Bell
9	Christopher Nolan	26	Hayao Miyazaki	43	Catherine Owens
10	Quentin Tarantino	27	Andrew Stanton	44	Charles Chaplin
11	Sam Mendes	28	Chan-wook Park	45	Damien Chazelle
12	Alfred Hitchcock	29	Fernando Meirelles	46	Jay Oliva
13	James Cameron	30	Florian Henckel von	47	John Blanchard
14	Stanley Kubrick	31	John Stockwell	48	Lee Unkrich
15	Wolfgang Petersen	32	Mel Gibson	49	Majid Majidi
16	Lana Wachowski	33	Michael Curtiz	50	Marius A. Markevici



1. director_name의 파생변수
train set에 나온 감독 이름을 추합하여 가장 높은 IMDb score를 낸 100개 영화를 만든 감독에 해당하면 YES 해당하지 않으면 NO를 부여

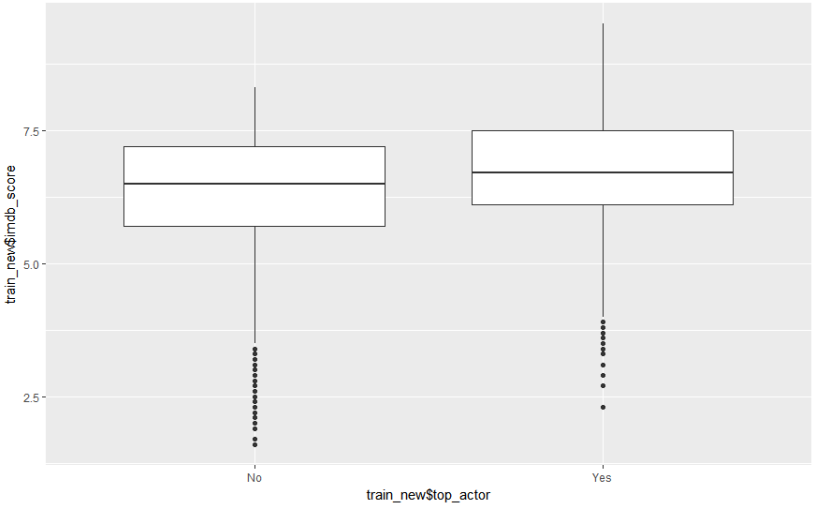
이 명단에 입각하여 test set에서도 해당 영화를 만든 감독이 명단에 있으면 YES 없으면 NO를 부여

2. Main Idea
명성이 있는 감독이 제작한 영화는 평점이 높을 것

3. 결측치 처리
100위 안에 드는 영화 중 감독의 이름이 없는 영화 19개: No 부여
즉, 정확히는 Top 100에 드는 81개 영화를 제작한 감독에게 Yes

04. 파생변수와 NA 처리

02/ top_actor



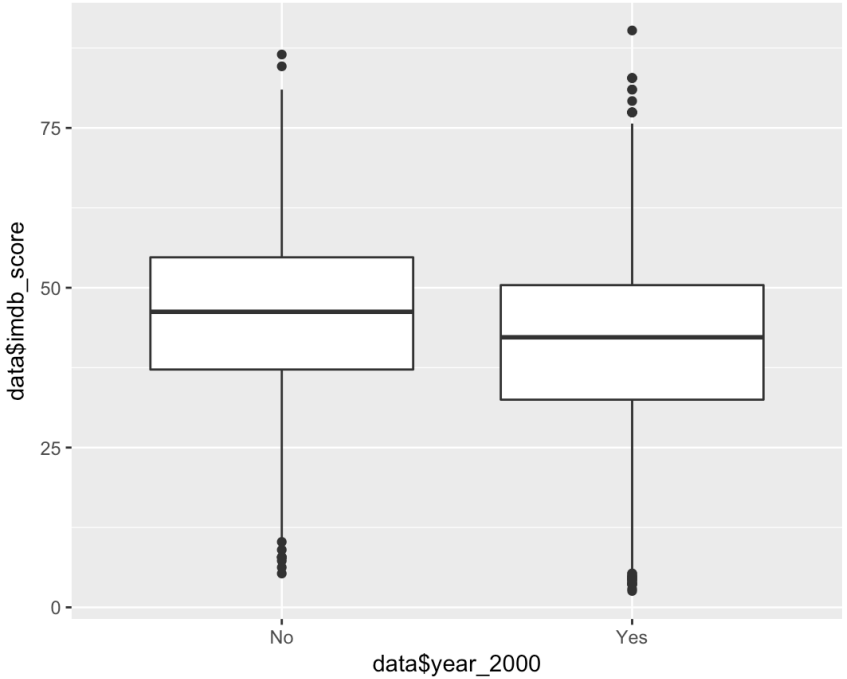
1. actor_name의 파생변수
train set에 나온 배우 이름을 수합하여 가장 높은 IMDb score를 낸 50개 출연한 배우가 actor 1, 2, 3 중 한 명이라도 있으면 YES(260명) 없으면 NO를 부여

2. Main Idea
유명하면서 과거 좋은 평점을 낸 영화에 출연한 배우가 나온다면 영화의 평점이 높을 것

3. 결측치 처리
결측치 없음

04. 파생변수와 NA 처리

03/ year_2000



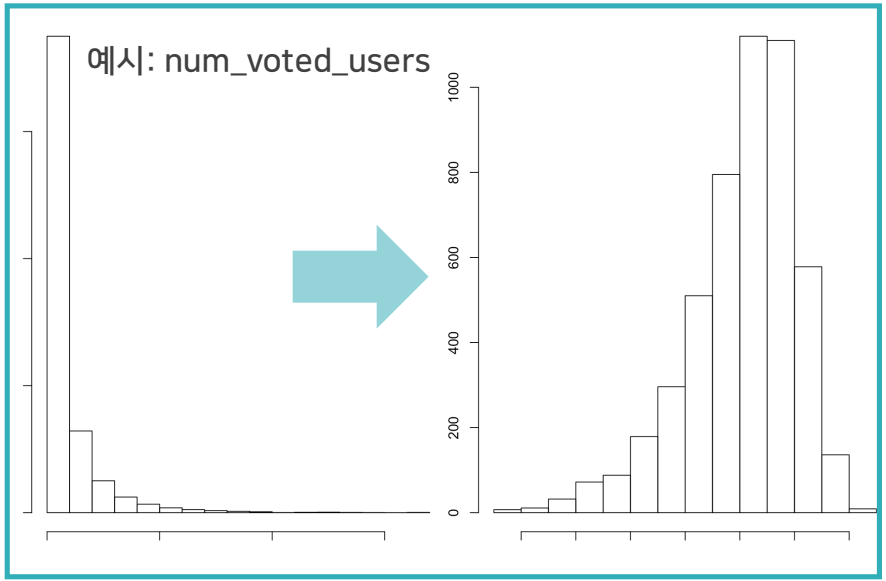
1. title_year의 파생변수
2000년 이후 영화이면 YES, 1999년 이전 영화이면 NO 부여

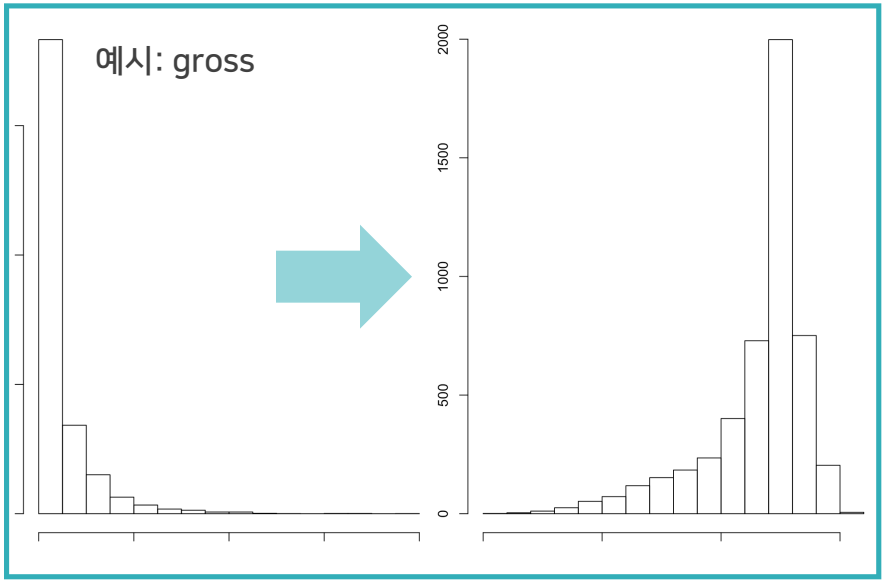
2. Main Idea
연도에 따라 유행하는 영화의 스타일이 있으며, 고전 영화의 경우 IMDb 온라인 영화 평가 사이트에 기록되려면 질적으로 뛰어난 영화일 것이라는 판단

3. 결측치 처리
median인 2005로 입력

05. 변수 transformation

log 변환: num_voted_users, gross, budget, num_critics_for_reviews, num_users_reviews, cast_total_fb_likes

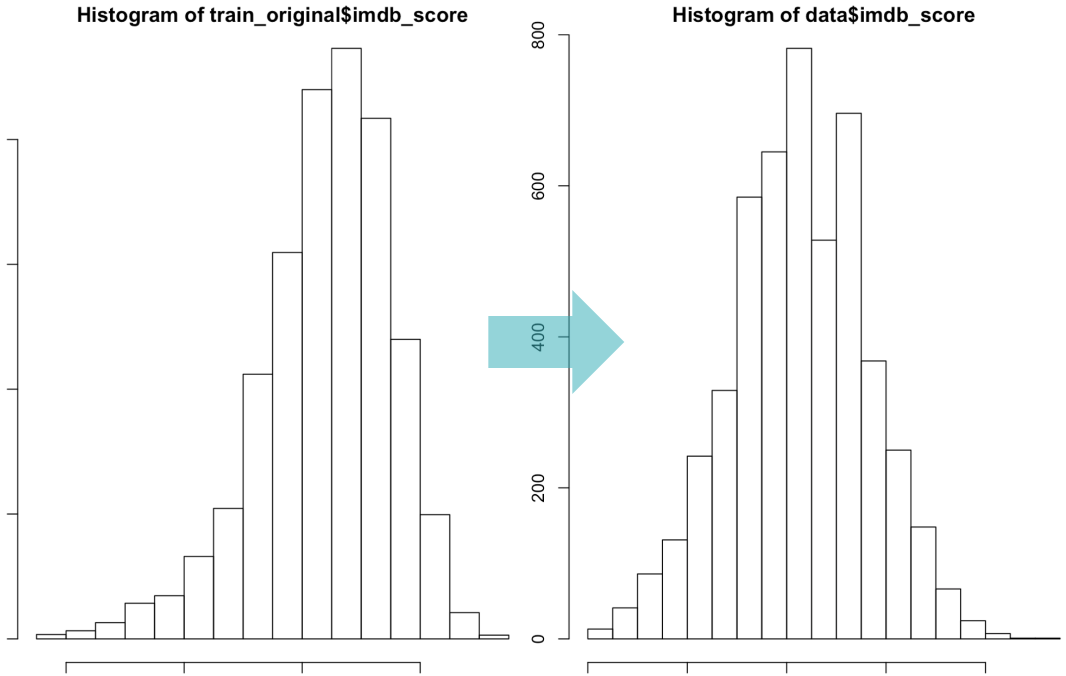




*budget 변수: outlier 관측치 6개(한국, 일본, 헝가리 영화)에 대해 달러로 표준화

**수치형 자료의 NA는 모두 mean 보다 robust한 median으로 처리

종속변수: Imdb_score 제곱하여 사용



기존의 데이터에 의하면 imdb score의 분포는 left tail이 긴 분포. 이를 좀 더 종 모양의 분포로 만들어주기 위하여 제곱하여 사용. 또한 notch를 통해 변수의 유의미함을 판단할 때, 자료의 대칭화 전제에 좀 더 알맞은 score 제곱을 사용.

최종 모델에 사용된 변수

color(C)
log_num_critic_for_reviews
duration(C)
genres(C)
log_num_voted_users
log_cast_total_facebook_likes
facenum_in_poster(C)
log_num_user_reviews

country(C)
content_rating(C)
log_budget
aspect_ratio(C)
Year2000(C)
top_actor(C)
top_director(C)
log_gross

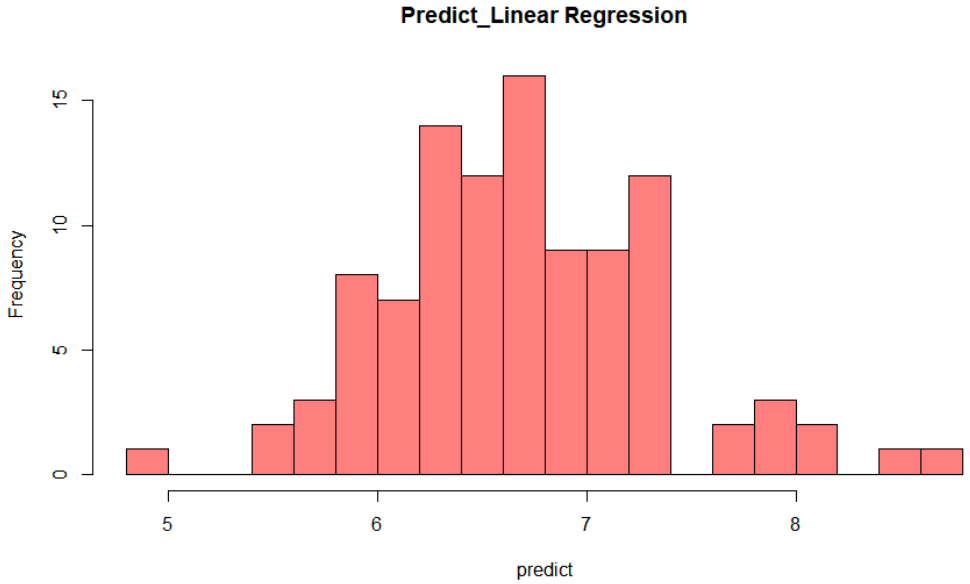
**(C)는 카테고리 변수를 의미. 나머지는 숫자형 변수

3개의 파생변수를 포함하여
총 16개 변수 사용

02. Model Selection

01. 선형회귀분석

```
Call:
lm(formula = movies_train$imdb_score ~ ., data = movies_train)

Residuals:
    Min      1Q  Median      3Q     Max
-42.390  -6.578   0.115   6.574  64.909

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                   50.21876    3.49933  14.351  < 2e-16 ***
colorColor                    -4.60836    0.81810  -5.633 1.87e-08 ***
log_num_critic_for_reviews     0.47478    0.25178   1.886  0.05939 .
durationmedium                -5.18949    0.41402 -12.534  < 2e-16 ***
durationshort                 -6.35527    0.54028 -11.763  < 2e-16 ***
genresAdventure                1.19798    0.74174   1.615  0.10636
genresAnimation                7.13655    0.97904   7.289 3.61e-13 ***
genresComedy                   1.86999    0.65779   2.843  0.00449 **
genresDrama                    9.29093    0.67859  13.692  < 2e-16 ***
genresOthers                   4.29713    0.91258   4.709 2.56e-06 ***
genresRomance                  2.85428    0.61923   4.609 4.14e-06 ***
genresSci-Fi                  -0.73366    0.69522  -1.055  0.29134
genresThriller                 0.87883    0.67632   1.299  0.19385
log_num_voted_users            3.78378    0.19922  18.993  < 2e-16 ***
log_cast_total_facebook_likes -0.98528    0.10573  -9.319  < 2e-16 ***
facenum_in_posterYes          -0.34739    0.32877  -1.057  0.29073
log_num_user_reviews          -1.26843    0.27421  -4.626 3.83e-06 ***
countryYes                    -3.44824    0.38773  -8.894  < 2e-16 ***
content_ratingNot Rated        4.26168    2.58841   1.646  0.09974 .
content_ratingOld              8.60142    2.81709   3.053  0.00228 **
content_ratingPG               2.66118    2.50016   1.064  0.28720
content_ratingPG-13            2.17277    2.48892   0.873  0.38272
content_ratingR                5.04249    2.46733   2.044  0.04104 *
log_budget                    -1.49909    0.13306 -11.266  < 2e-16 ***
aspect_ratio                  -1.58517    0.61234  -2.589  0.00966 **
year_2000Yes                  -1.65350    0.40568  -4.076 4.66e-05 ***
top_actorYes                   2.84253    0.44475   6.391 1.80e-10 ***
top_directorYes                8.70616    0.71816  12.123  < 2e-16 ***
log_gross                     -0.03576    0.08829  -0.405  0.68548
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.85 on 4914 degrees of freedom
Multiple R-squared:  0.375,     Adjusted R-squared:  0.3715
F-statistic: 105.3 on 28 and 4914 DF,  p-value: < 2.2e-16
```



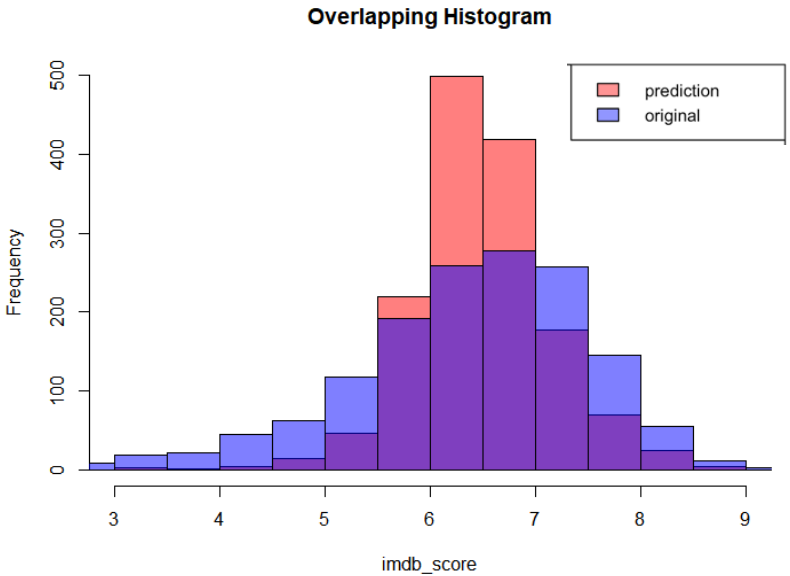
(좌) linear regression output

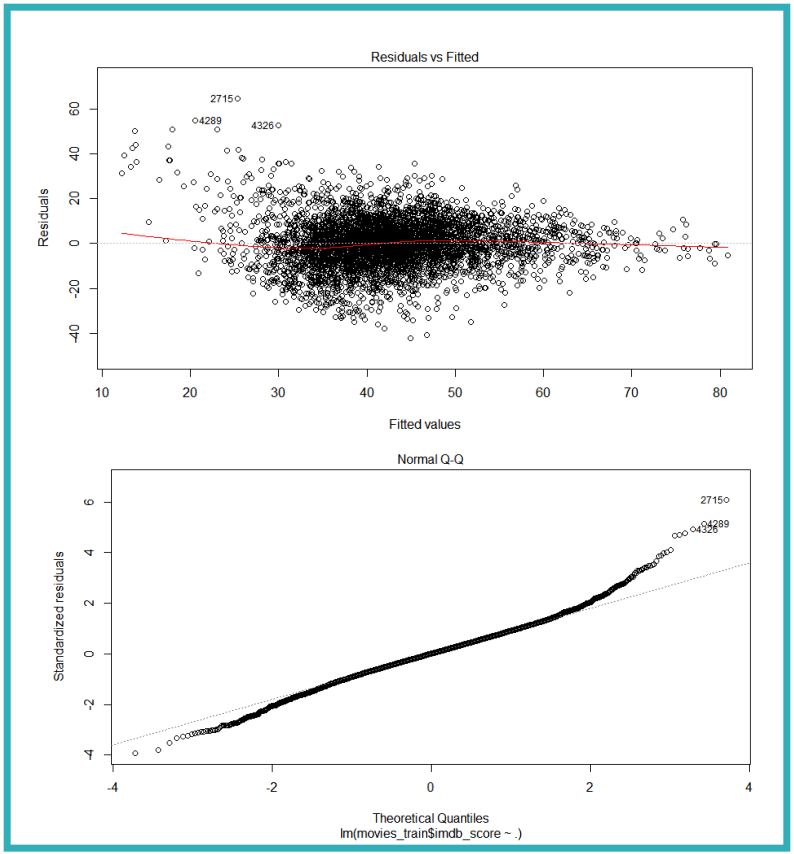
(위) prediction on test set of 100 obs.

R-squared: 0.375

test mse: 0.940

01. 선형회귀분석



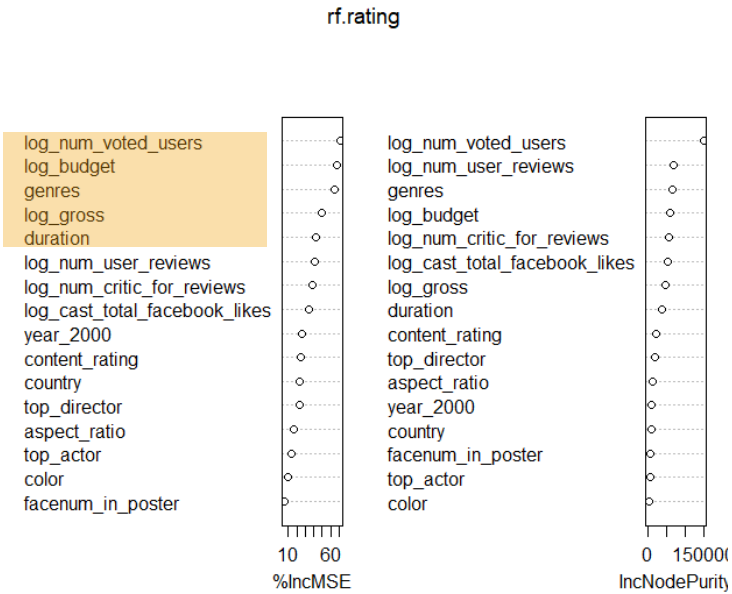


(우) 선형회귀모델의 잔차그림과 qqplpt

(위) 앞에서 활용한 모델을 가지고 예측된 값과 원래의 imdb score를 비교하여 히스토그램 그린 것

02. 랜덤 포레스트

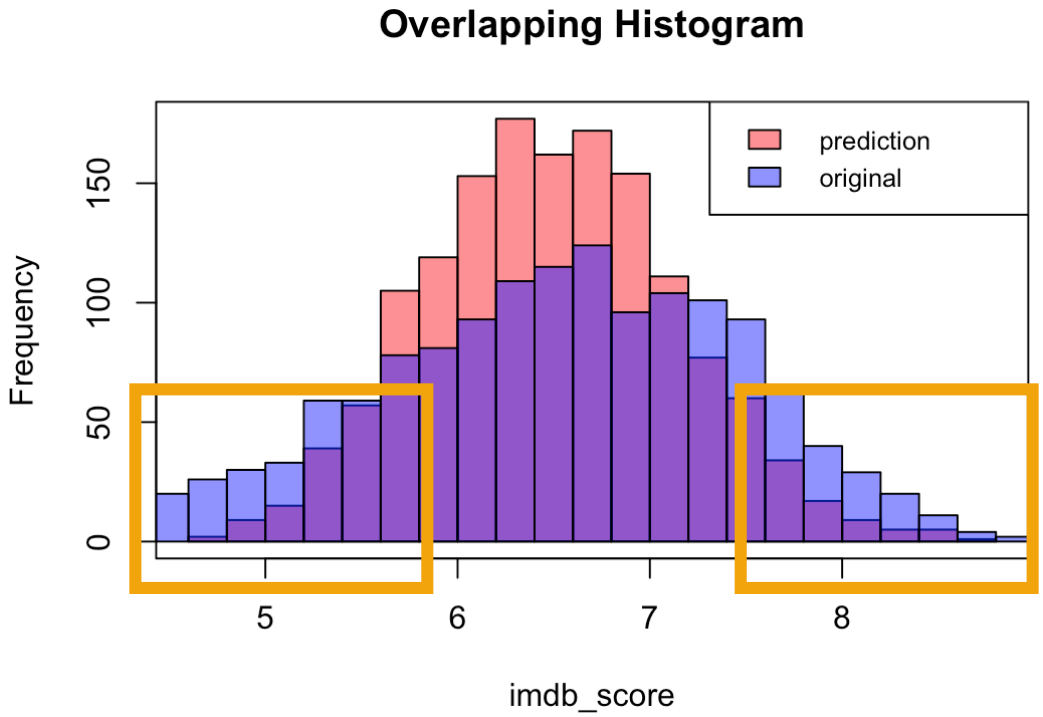
```
#random forest
rf.rating=randomForest(imdb_score~.,data=dat[train,],mtry=6,importance=TRUE)
yhat.rf=predict(rf.rating,newdata=dat[test,])

#test mse
test.mse=mean((sqrt(dat[test,]$imdb_score)-sqrt(yhat.rf))^2)
```



- Cross Validation으로 주어진 train set에서 7:3의 비율로 train set과 test set을 분배. 트리 5000 개를 이용하고, 설명변수가 16개인 점을 고려해 6개의 변수를 매 트리에서 사용. (test mse = 0.671)

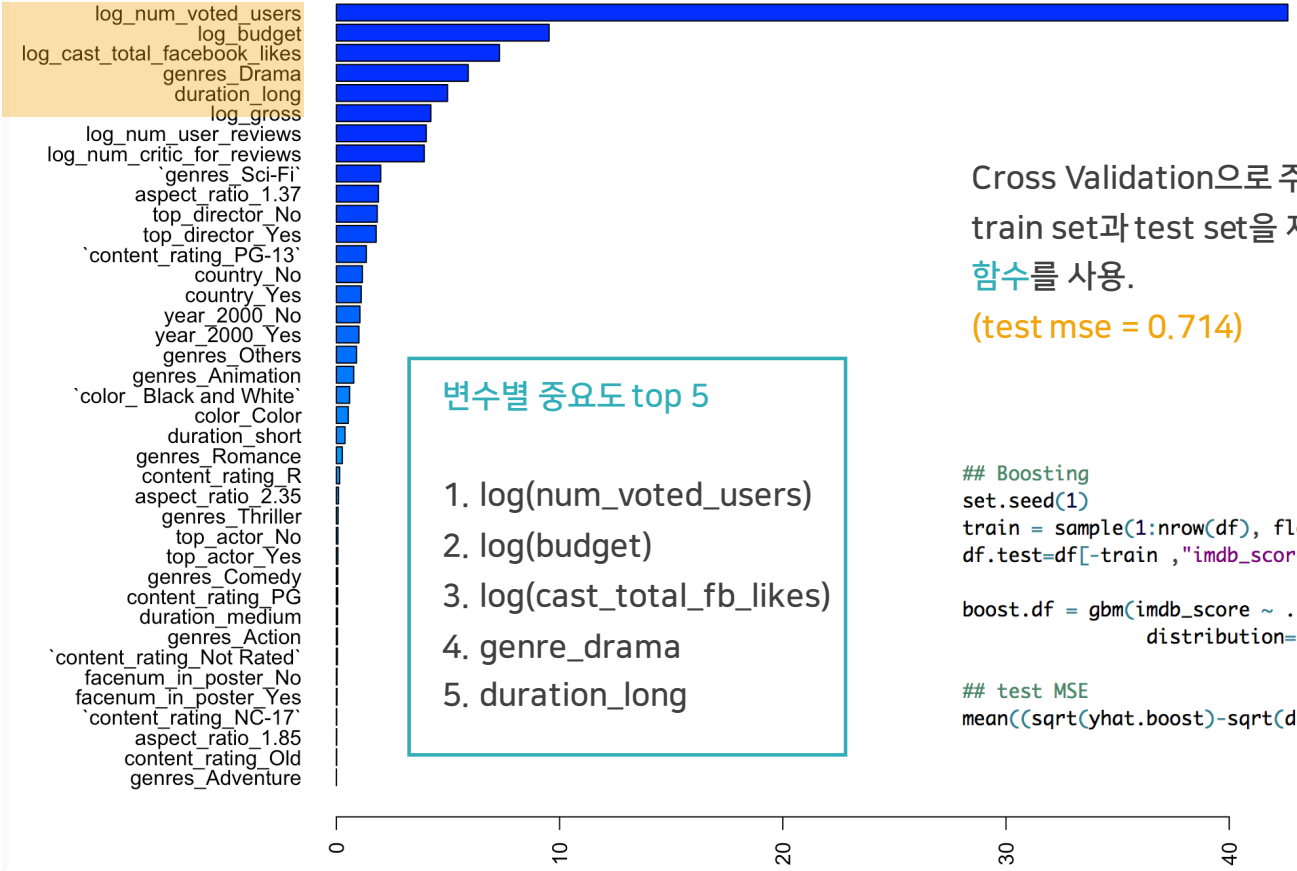
- 변수별 중요도 top 5
  1. log(num_voted_users)
  2. log(budget)
  3. genres
  4. log(gross)
  5. duration

02. 랜덤 포레스트



- Train set을 다시 7:3의 비율로 train, test를 나눔. 70%의 train set으로 만든 모델을 30%의 test set에 대해 prediction 한 것 (붉은 색)과 원래의 imdb_score (푸른 색)를 비교한 것.
- Original데이터의 분포와 비교해 볼 때, 양끝을 잘 맞추지 못하는 경향이 있다. Original 데이터의 분포가 6과 7을 중심으로 형성되어 있기 때문에 랜덤 포레스트 모델 또한 6과 7사이 값을 prediction으로 산출하는 경향.

03. 부스팅



변수별 중요도 top 5

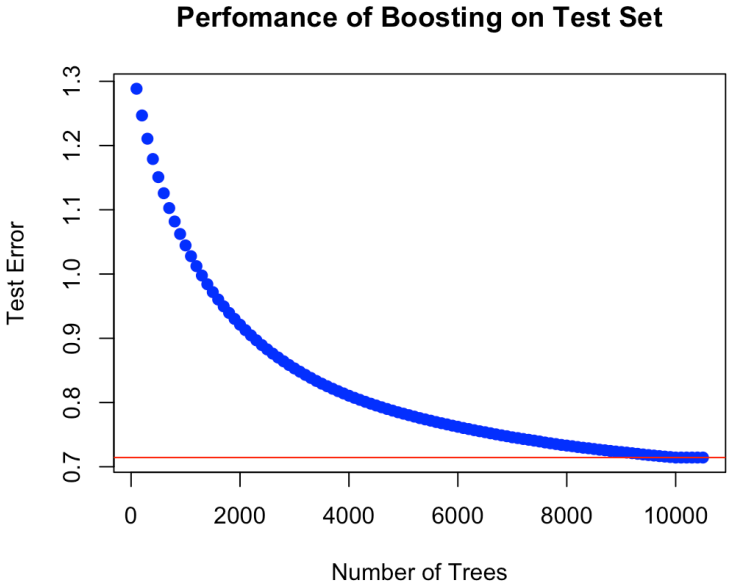
1. log(num_voted_users)
2. log(budget)
3. log(cast_total_fb_likes)
4. genre_drama
5. duration_long

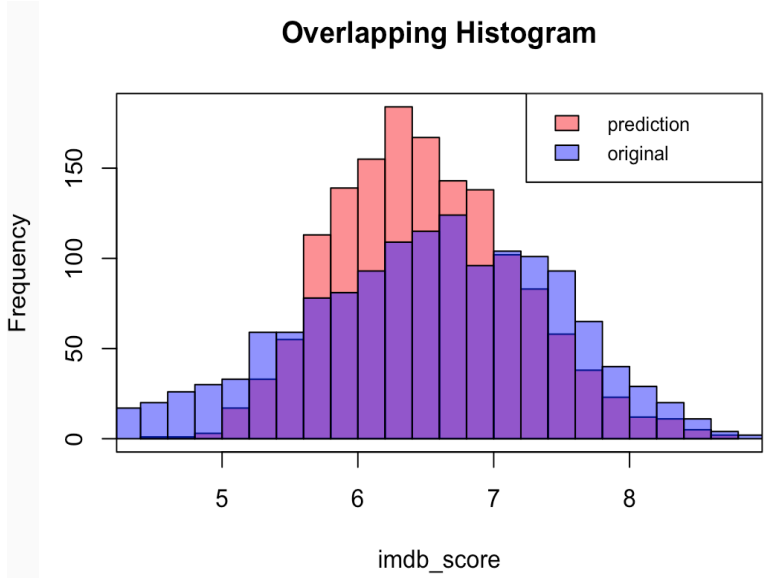
Cross Validation으로 주어진 train set에서 다시 7:3의 비율로 train set과 test set을 재분배. 트리를 5000개 이용하여 gbm() 함수를 사용.

(test mse = 0.714)

```
## Boosting
set.seed(1)
train = sample(1:nrow(df), floor(nrow(df)*0.7))
df.test=df[-train ,"imdb_score"]

boost.df = gbm(imdb_score ~ ., data = df[train, ],
               distribution="gaussian",n.trees=10000, interaction.depth=4)

## test MSE
mean((sqrt(yhat.boost)-sqrt(df.test))^2) #0.7136798
```

03. 부스팅

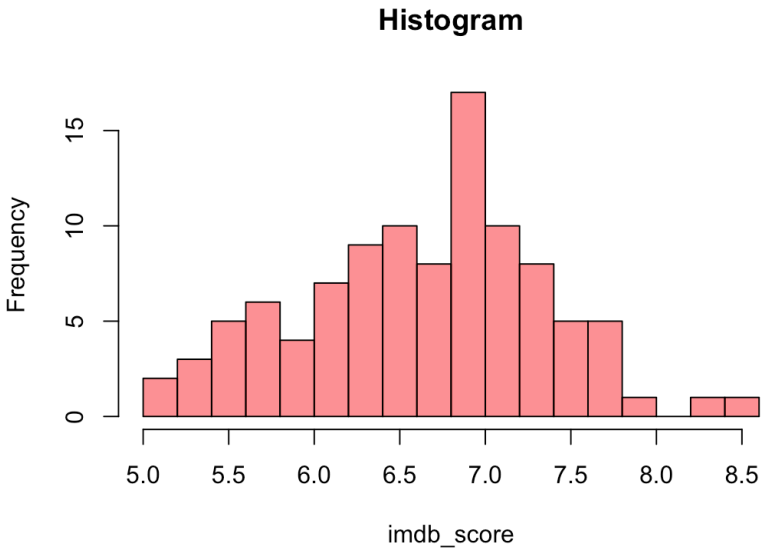




(좌) 트리 수에 따른 test error의 변화. Tree 수가 10000개 이상되는 시점 부터 부스팅의 test mse가 랜덤 포레스트 결과로 산출된 test mse(붉은 선)에 수렴한다. 랜덤 포레스트보다 우월한 결과를 내지 못한다.
(우) 앞에서 활용한 모델을 가지고 예측된 값과 원래의 imdb score를 비교하여 히스토그램 그린 것

03. 최종 모델 선택 및 결과

Model Selection



test mse가 가장 낮은 랜덤 포레스트 모델을 최종 모델으로 선정. 위 히스토그램은 주어진 100개 case에 최종 모델로 prediction 한 것.

회귀분석
test mse=0.940

랜덤 포레스트
test mse=0.671

부스팅
test mse=0.714

GBM(Gradient Boosting)은 다양한 parameter를 사용

1. n.trees: 트리의 수. 값이 증가할 수록 training set의 에러를 줄이지만, over-fitting의 우려가 있다.
2. Interaction.dept: maximum nodes per tree로 하나의 트리를 생성할 때 최대로 사용될 수 있는 split의 수
3. Shrinkage: learning rate로 값이 클 수록 모델이 천천히 학습한다. 값이 크게 되면 over-fitting의 우려가 있다.
4. N.minobsinnode: 트리의 터미널 노드의 최소 관측치 수
5. …

각각 parameter에 대해 cross-validation 혹은 caret 패키지를 사용하여 최적의 값을 찾으면 boosting 성능이 향상될 것.

Why Random Forests over Boosting?

다양한 부스팅 알고리즘

1. AdaBoost: 다수결을 통한 정답 분류 및 오답에 가중치 부여
2. GBM: Gradient Descent 알고리즘으로 오답에 가중치 부여
3. xgboost: GBM대비 성능 향상 (2014년 공개)
4. light GBM: xgboost 대비 성능 향상 및 자원소모 최소화, 대용량 데이터 학습 가능 (2016년 공개)

04. Further Comments: 한계점

“Further Comments”

01/ 데이터 자체의 문제점

크롤링으로 얻어진 데이터. 잘못 인코딩 된 것 부터 최근 자료로 업데이트 되지 않은 자료.

02/ 주어진 변수들의 imbalance

주어진 변수 데이터에서 카테고리 데이터의 경우 카테고리별 data 수의 imbalance가 심함.

03/ 모델의 개선점

모델을 선정하는 과정에서 imdb score 데이터가 보통 4~7(10점 스케일) 사이의 값에 쏠려있어 이 범위를 벗어난 것에 대해서는 예측력이 약함.

Thank You

Any Question?